Lab Week 8 Part 1 – Instructions

Note: The tasks described herein are in addition to the tasks described in "Description". Do these after completing (or concurrently with) the tasks described in "Description" .

Before setting up a Watchdog Timer (WDT) one needs a good idea of the normal execution time of the program. Then one sets the timeout of the WDT to be a little more than the normal execution time so the WDT allows the program to execute normally without any intervention. The WDT springs into action and resets the MSP432 only if the execution time of the program exceeds the timeout of the WDT due to abnormal execution.

In the case of this project, the program is an image processing algorithm which blurs a 2D image which is well explained in '**Description**'. This program can be found in the zip file '**program_monitor.zip**.' When you extract the zipped file you will find that it has include files blurfilter.c and header file blurfilter.h The dimension of the matrix is initially set at 75X75 in the blurfilter.h file.(Note: the dimension of the matrix is in the header file, not in main.c). To find the execution time of the matrix, code similar to the one developed in Lab Week 6 Part 2 (execution time matrix multiplication) has been given in '**Image_Proc_Exec_Time.zip**.' This program has been given only to save you time; understand it well before executing it. Your first task is to run the code for matrices of size 20X20, 40X40, 60X60 and 80X80 at a CPU clock (MCLK) of 12 MHZ and fill up the Table # 1below. Keep MCLK constant at 12 MHZ

Dimension of Matrix	Exec Time in Secs
20 X 20	0.0242
40 X 40	0.0982
60 X 60	0.222
80 X 80	0.395

Now calculate the timeout of WDT for different values of the divider of the clock (ACLK set at 128 KHZ thru REFOCLK) and fill up the Table #2 below

Divider / Iterations	512	8192	32K
1	0.004	0.064	0.25
2	0.008	0.128	0.5
4	0.016	0.256	1.0
8	0.032	0.512	2.0

Choose the divider and No of Iterations in such a way that the timeout is greater than some of the execution times in the first table and is less than the some of the execution times in the first table. In other words, matrices of smaller dimension complete their image blurring without any hindrance from WDT , whereas matrices with bigger dimension will face a WDT reset and will not be able to complete the image blurring in the set amount of time.

The last step is to develop the code incorporating WDT with the image processing code. To this end, Make a copy of **Image_Proc_Exec_Time** project and rename it as Lab_Week8_Part1 or something similar. Remove all the code related to calculation of elapsed time ,almost all except the two lines below

```
acquireImage (&in, MIN_VALUE, MAX_VALUE, SIZE); // Acquire image
blurFilter (&in, &out); // Process image
```

and replace the deleted lines with WDT related code. For the latter you can refer either to '**wdt_a_service_the_dog'** or '**Watchdog_Timer_Example'**. LED2 should be initially lit RED and if the code executes to completion without intervention from WDT it should turn GREEN until it is eventually reset by the WDT. If the dimension of the matrix is larger execution will not complete and LED2 will remain RED due to repeated WDT resets. For the checkoff you have to start with small dimensions of the matrix and increase the dimension gradually until the LED2 no longer turns GREEN and remains RED. Determine the largest size of matrix which can be completed without hitting a WDT reset for 4 different CPU clock frequencies 12 MHZ, 6 MHZ,3 MHZ and 1.5 MHZ (obtained by varying the MCLK divider) and 2 different WDT timeouts 0.25 sec and 0.5 sec (obtained by varying ACLK and/or no. of clock iterations for WDT.

Fill up the Table #3 below and upload as part of your submission

WDT Timeout/CPU Clock	12 MHZ	6 MHZ	3 MHZ	1.5 MHZ
0.25 sec	64	45	32	22
0.5 sec	90	64	45	32